VHDL2TV: Outil de génération des programmes prêts à simuler

Manuel d'utilisation

Equipe System On Chip (SOC) - Laboratoire d'Informatique de PARIS 6 (LIP6)
Unité Mixte de Recherche - UMR 7606 (CNRS - UPMC)

Abdelrezzak BARA,
Emmanuelle ENCRENAZ

12 Janvier 2011

Résumé : Ce document est un guide d'utilisation de l'outil VHDL2TV (Translation of VHDL Programs to Timed Automata) qui permet de générer un programme VHDL temporisé prêt à simuler à partir d'un simple programme VHDL décrivant la fonctionnalité du circuit à analyser, les annotations temporelles associées à ses portes et l'environnement de ses signaux d'entrée. Ceci permet de faciliter la tâche d'analyse de circuits en utilisant les techniques de simulation. L'outil VHDL2TV a été utilisé pour passer des simulations sur la mémoire SPSMALL [CEFX06].

Le module de génération des programmes VHDL temporisés de cet outil fait partie des modules de l'outil VHDL2TA.

Mots clés : Circuits asynchrones, Circuits Mémoires, Langage VHDL, Programmes VHDL temporisés, Abstraction fonctionnelle (MYGALE), Abstraction temporelle (TIMEX), Architecture de la mémoire SPSMALL, Simulation, ModelSim.

Tables des matières

1. Introduction 03

2. Description de VHDL2TV 04

2.1. Architecture 04

2.2. Les descriptions d'entrée sortie 04

3. Mode d'emploi de VHDL2TV 06

3.1. Installation 06

3.2. Les descriptions d'entrée de VHDL2TV 06

3.2. Les commandes d'appel de VHDL2TV 07

4. Exemples de tests 10

Porte n-xor 10

5. Conclusion 14

Références 15

1. Introduction

Ces travaux concernent l'analyse de circuits numériques temporisés. Ces circuits sont composés de portes logiques auxquelles des délais de propagation de fronts sont associés. Ces circuits sont représentés à la fois par un modèle fonctionnel décrit en VHDL et par un modèle temporel qui associe à chaque bloc fonctionnel des délais de propagation. Afin de l'analyser, on intègre les annotations temporelles du modèle temporel dans le modèle fonctionnel afin d'obtenir un modèle fonctionnel et temporel décrit en VHDL sur lequel les techniques de simulation classique peuvent être appliquées.

Pour faciliter une telle analyse, il serait pratique d'avoir un outil qui permet de générer un programme VHDL temporisé prêt à simuler à partir d'un simple programme VHDL décrivant un circuit donné et les annotations temporelles associés aux portes (les délais de propagation des fronts des portes) du circuit. Ceci est l'objet de l'outil VHDL2TV.

L'outil VHDL2TV a été développé au LIP6, Université de Pierre et Marie Curie (Paris 6), dans le cadre du projet ANR VALMEM. Son code binaire ainsi que quelques exemples de tests de circuits sont téléchargeables à partir de la page web référencée [BE11c].

Organisation du manuel : la deuxième section de ce manuel décrit l'architecture de l'outil et ses caractéristiques. Dans la section 4, on présente le mode d'installation et d'emploi de l'outil. La section 4 est consacrée à la présentation de quelques exemples de simulation de circuits en utilisant : l'outil VHDL2TV (pour générer le programme VHDL temporisé prêt à simuler) et l'outil de simulation ModelSim [MS] (pour simuler ce programme). La section 5 conclue ce document.

2. Description de VHDL2TV

Dans cette partie, on présente l'architecture de l'outil, ses modules et ses descriptions d'entrée/sortie.

2.1. Architecture

L'architecture de l'outil est présentée dans la figure 1, mentionnée ci-dessous.

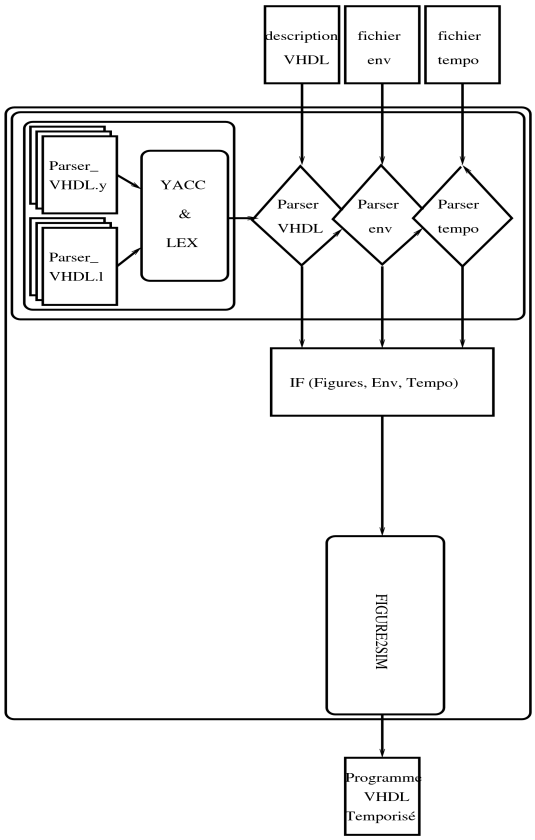


Figure 1 : l'architecture de l'outil VHDL2TV.

Principalement, VHDL2TV intègre :

1. un analyseur qui permet de donner une représentation structurelle aux descriptions d'entrée de l'outil (qui sont syntaxiquement correctes) : la description fonctionnelle du circuit en VHDL, la description temporelle des portes du circuit et la description d'environnement des signaux d'entrée (cet analyseur est le même que celui de l'outil VHDL2TA).

2. Un module de génération des fichiers VHDL qui constituent ensemble le programme VHDL temporisé prêt à simuler, à partir du format intermédiaire produit par l'analyseur (Ce module nommé *`figure2sim`* est intégré dans l'outil VHDL2TA).

2.2. Les descriptions d'entrée et sortie

Comme le montre la figure 4, l'outil prend en entrée les descriptions suivantes :

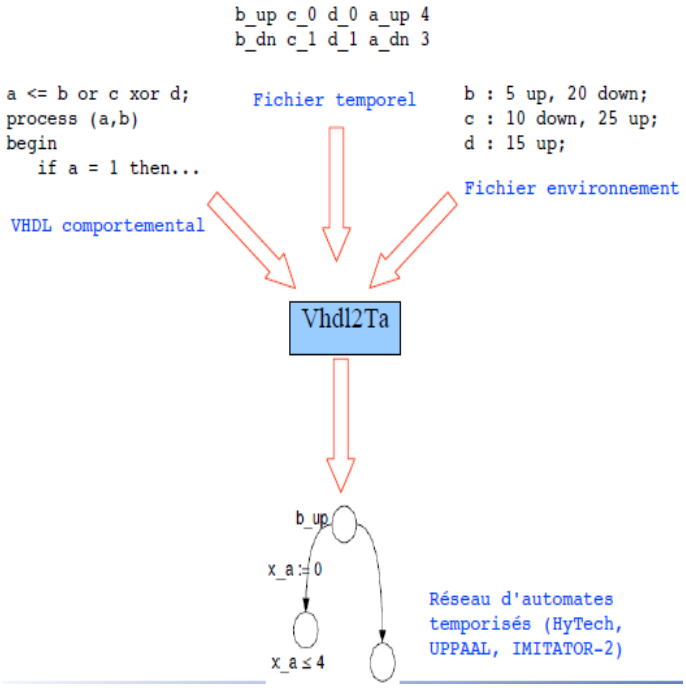


Figure 2 : Les descriptions d'entrée / sortie de l'outil VHDL2TA.

- **la partie fonctionnelle du circuit (programme VHDL)** est donnée par une description en VHDL. Chaque bloc combinatoire (resp. séquentiel) est décrit par une affectation concurrente (resp. processus).

- **la partie temporelle du circuit (délais des portes)** décrit les délais de propagation des fronts sur la sortie des portes du circuit. Il existe deux types de descriptions : 1) les descriptions données sous forme d'un fichier dans lequel deux intervalles de délais sont associés à chaque porte (modèle temporel bi-bounded delay). 2) les descriptions dans lesquelles on associe un fichier de timing (configuration, front-sortie, temps) pour chaque porte (modèle STG complet). On leur ajoute aussi un fichier principal dans lequel on associe à chaque porte le nom (chemin) du fichier de timing correspondant. On note que dans ce modèle, les descriptions sont représentées par ces fichiers principaux.

- **l'environnement du circuit (Fichier d'environnement)** décrit le comportement des signaux d'entrée.

L'outil VHDL2TV génère, ensuite, le programme VHDL temporisés prêt à simuler.

3. Mode d'emploi de VHDL2TV:

3.1. Installation.

L'outil est disponible en téléchargement sur le site web dont l'URL est donnée ci-dessous :
http://www-asim.lip6.fr/~ema/valmem/vhdl2tv/index.html.

3.2. Les descriptions d'entrée de VHDL2TV.

Dans cette partie, on définit la format des descriptions d'entrée de l'outil employées pour générer le modèle d'automates temporisés représentant un circuit donnée. On se sert de la porte n-xor comme un exemple, dont la sortie est nommée s et les deux entrées sont nommées e1 et e2, respectivement.

3.2.1. La description fonctionnelle du circuit est un programme en VHDL, où les blocs combinatoires du circuit sont décrits par des affectations concurrentes et les blocs séquentiels sont décrits par des processus. Les programmes acceptés par l'analyseur de l'outil ne contiennent que des affectations simples et des processus de type If-elseif . . . -elseif {else} qui affectent une nouvelle valeur à un seul signal donné. Le format de ces affectations et de ces processus est décrit ci-dessous. Le fragment de la grammaire VHDL, qui génère ce sous ensemble du langage VHDL, est donné dans l'annexe B.1.

```
1)      s <= f (e_1, . . . , e_m).
2)      Process_Name Process (e_1, . . . , e_m)
        begin
        .       if guard_1
        .               then s <= f_1 () ;
        .       else-if guard_2
        .               then s <= f_2 () ;
        .       .
        .       .
        .       else-if guard_n
        .               then s <= f_n () ;
        end
```

La description de la porte n-xor en VHDL est donné comme suit :

```
-- Entity Declaration
ENTITY n_xor IS
  PORT (
         s : out   BIT;
         e_1 : in    BIT;
         e_2 : in    BIT
  );
END n_xor;
```

```
-- Architecture Declaration
ARCHITECTURE arc OF n_xor IS

BEGIN
  s <= not (e_1 xor e_2);
END arc;
```

3.2.2. La partie temporelle du circuit (délais des portes) décrit les délais de propagation des fronts montants et descendants des signaux sur les portes du circuit. Généralement, la description est donnée sous forme d'un fichier dans lequel deux intervalles de délais (ou un délai ponctuel) sont associés respectivement aux fronts montants et descendants de chaque porte (modèle *temporel bi-bounded delay*).

```
-- Delai pour chaque front
  s : [1,2]- , [3;4]+ ;
```

Une autre option consiste à associer un fichier de timing pour chaque porte (modèle STG complet). Dans ces fichiers, on associe un délai ponctuel pour chaque configuration et pour chaque front d'un signal d'entrée d'une porte qui induit un changement sur la sortie. Un exemple d'un modèle temporel complet pour la porte n-xor est donné ci-dessous. On peut remarquer que le premier modèle (modèle *temporel bi-bounded delay*) est une abstraction de ce dernier modèle.

```
--     Delai     pour     chaque
configuration
  e_1 e_2 s time
  0   u   d  2ns
  0   d   u  3ns
  u   0   d  2ns
  d   0   u  3ns
  1   d   d  1ns
  1   u   u  4ns
  d   1   d  1ns
  u   1   u  4ns
```

La syntaxe pour les deux types de descriptions temporelles est décrite par la grammaire donnée dans l'annexe B.2.

3.2.3. La description d'environnement est donnée sous forme d'un fichier qui contient une liste de comportements associés aux signaux d'entrée du circuit à modéliser. Un comportement d'un signal est une suite alternée de fronts montants et descendants à des instants croissants (ou intervalle d'instants). La syntaxe des descriptions d'environnement est décrite par une grammaire génératrice donnée dans l'annexe B.3.

Voici un exemple de description d'environnement des signaux d'entrée de la porte n-xor :

```
-- Env
e_1 : 5 up, 15 down;
e_2 : 10 up, 20 down;
```

3.3. Les commandes d'appel de VHDL2TV

La syntaxe générale des commandes qu'on passe sur l'outil est donnée comme suit :

```
vhdl2tv <vhld_file> [-temp <temp_file> temp_type] [-env env_file]
[-out testbench_file] [sim_options] [options]
```

Les paramètres passés dans cette commande :

`vhdl_file` : le fichier VHDL qui décrit la fonctionnalité du circuit. Ce paramètre est toujours obligatoire.

`-temp` (abrégé à `-t`) : c'est l'option qui permet de décrire la partie temporelle du circuit. Elle prend

en paramètre, le fichier temporel `temp_file` et son type `temp_type` (1 s'il s'agit d'un modèle temporel complet, 2 s'il s'agit d'un modèle *temporel bi-bounded delay*). Par défaut, les délais des portes sont nuls, si cette option n'est pas spécifiée. On peut mettre `-t` à la place de `-temp` dans la commande.

`-env` (abrégé à `-e`) : cette option décrit l'environnement du circuit. Elle permet de fournir le fichier d'environnement `env_file`. Par défaut, le comportement des signaux d'entrée du circuit est stable à la valeur 0, si cette option n'est pas passée en paramètres de la commande `vhdl2ta`.

`-out` (abrégé à `-o`) : cette option permet de passer en paramètre le nom du fichier testbench qui constitue, avec les autres fichiers VHDL décrivant la partie fonctionnelle et temporelle des portes du circuit, le programme prêt à simuler. Tous les fichiers *.vhd* générés sont dans le même répertoire courant que celui du fichier testbench.

Les options (sim`_options`) spécifiques au programme prêt à simuler à générer sont :

`-sim_version` (abrégé à `-simv`) : il existe deux versions des programmes VHDL prêts à simuler qu'on peut générer avec l'outil VHDL2TA. **1)** la première version dans laquelle on associe un seul composant pour tout le circuit (le programme n'est constitué que de deux fichiers). **2)** la deuxième version dans laquelle un composant VHDL est associé pour chaque porte du circuit en décrivant sa fonctionnalité et ses annotations temporelles (en conséquence, le programme est constitué d'un fichier testbench et de plusieurs fichiers VHDL). Par défaut, l'option prend la valeur `gate` représentant la deuxième version. A titre d'indication, la première version est représenté par la valeur `circuit`.

`-sim_format` (abrégé à `-simf`) : indique le format VHDL dans lequel les composants associés aux portes du circuit sont représentés. Les composants peuvent être représentés soit par des processus (`process`) soit par des affectations concurrentes (`assign`). Il est très recommandé d'employer la première représentation pour différents raisons car elle est bien adaptée pour représenter les blocs séquentiels et elle permet une meilleure lisibilité du code. Cette option prend la première valeur, par défaut. A titre d'indication, on ne peut employer le deuxième format (`-simf = assign`) que dans le cas où le modèle temporel passé en paramètre de l'option `-temp` est de type 1 (modèle temporel complet).

`-temp_select` (abrégé à `-ts`) : cette option peut être employée quand l'option `-temp` prend en paramètre un modèle temporel de type 2 (fichier de délais). Avec cette option, on peut spécifier les critères de sélection des délais des configurations sur un intervalle de délais. Les critères de sélection que cette option peut prendre en paramètre sont listés ci-dessous :

`min` : sélectionner les bornes inférieures des intervalles des délais associées aux portes.
`max` : sélectionner les bornes supérieures.
`moy` : sélectionner les valeurs moyens (leur partie entière supérieure) des bornes des intervalles.
`random` : sélectionner des délais aléatoires dans les intervalles des délais (c'est la valeur que `temp_select` prend par défaut).

Le reste des options (`[options]`) qu'on peut passer en paramètres dans la commande `vhdl2tv` sont donnés ci-dessous.

-debug (abrégé à -d) : cette option permet de donner plus d'informations de debugage.

-warning (abrégé à -w) : cette option permet d'afficher tous les warnings sans exception.

A titre d'indication pour avoir un menu d'aide de la syntaxe de la commande vhdl2tv., il suffit de taper la commande suivante :

```
vhdl2ta -h (-help ou --help).
```

Exemples d'appels

La commande `vhdl2ta n_xor.vhd -t n_xor_delays.tmp 2 -e n_xor.env -o n_xor_testbench.vhd -ts max` génère le programme VHDL prêt à simuler dont le fichier testbench est nommé `n_xor_testbench.vhd` à partir des intervalles des délais des portes.

La commande `vhdl2ta n_xor.vhd -t n_xor_delays.tmp 2 -e n_xor.env -o n_xor_testbench.vhd` génère le programme VHDL prêt à simuler dont le fichier testbench est nommé `n_xor_testbench.vhd` à partir des intervalles des délais des portes en considérant cette fois que les bornes supérieurs.

La commande `vhdl2tv n_xor.vhd -t n_xor_timing.tmp 1 -e n_xor.env -o n_xor_testbench.vhd` génère le programme VHDL prêt à simuler dont le fichier testbench est nommé `n_xor_testbench.vhd` à partir des délais de configurations.

La commande `vhdl2tv d_flip_flop.vhd -t d_flip_flop_timing.tmp 1 -e d_flip_flop.env -o d_flip_flop_testbench.vhd -vsim circuit` génère la première version du programme VHDL prêt à simuler (par défaut `-simv = gate`). On rappel que ce programme est constitué que de deux fichiers .vhd : le fichier qui décrit le composant associé au circuit et le fichier testbench `d_flip_flop_testbench.vhd`.

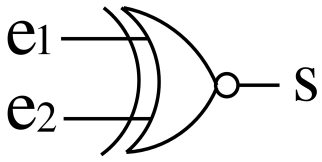
La commande `vhdl2tv d_flip_flop.vhd -t d_flip_flop_timing.tmp 1 -e d_flip_flop.env -o d_flip_flop_testbench.vhd -simf assign` génère le programme VHDL prêt à simuler dont les composants associés aux portes du circuit sont représentés par des affectations concurrentes (par défaut, les composants sont représentés par des processus `-simv` = process).

4. Exemples de tests

Dans cette section, on présente un exemple de tests pour l'outil VHDL2TV. L'intégralité des tests déroulés sur divers circuits est décrite dans la page web la page web de l'outil référencée [BE11c].

Porte n-xor :

la représentation de la porte logique n-xor (OU-exclusif complémenté) dont les entrées sont *e1* et *e2* et la sortie est *s* et sa fonctionnalité sont décrites dans la figure 3 présentée ci-dessous.



(a)

e_1	e_2	$\overline{e_1 \oplus e_2}$
0	0	1
0	1	0
1	0	0
1	1	1

(b)

Figure 3 : Porte n-xor : a) son symbole**.** b) sa table de vérité.

Nous avons vu dans la section précédente :

–le programme VHDL qui décrit la fonctionnalité de la porte.
–le fichier d'environnement qui décrit le comportement des deux signaux entrées *e1* et *e2*.
–le fichier de timing qui contient les délais des configurations de la porte.
–le fichier d'intervalles des délais associés aux fronts montants et descendants de la porte.

Nous avons fait deux tests. Dans le premier, on emploi le fichier d'intervalles de délais tandis que dans le deuxième on emploi le fichier de timing. Dans les deux tests, nous avons effectué l'analyse par simulation en utilisant le simulateur ModelSim sur le programme VHDL prêt à simuler générée par l'outil VHDL2TV. Le but est de vérifier si le comportement de la sortie de la porte est conforme à sa fonctionnalité.

A titre d'indication, ModelSim est un outil de simulation HDL de Mentor Graphics qui supporte plusieurs langages : Verilog, SystemVerilog, VHDL, SystemC.

Premier test :

Le programme VHDL généré par l'outil VHDL2TV est constitué de deux fichiers VHDL *:* un fichier qui décrit la porte n-xor, sous forme d'un composant, dans lequel les informations temporelles fourni dans le fichier de délais ont été intégrés (on a choisi d'intégrer les valeurs maximales quand il s'agit d'un intervalle pour un front, car c'est un modèle ponctuel), un fichier testbench qui décrit l'environnement dans un process et instancie les ports du composant n-xor par les signaux *e1*, *e2* et *s*. Les deux fichiers sont donnés dans l'annexe A.1.

La commande employée pour générer ce programme VHDL temporisé est donnée comme suit :

```
vhdl2tv n_xor.vhd -t delay.tmp 2 -e env1.env -o testbench.vhd -ts max
```

A titre d'indication, on peut employer l'outil VHDL2TA en utilisant son mode d'emploi `sim` pour générer ce programme VHDL temporisé. La commande employée dans ce cas est donnée comme suit :

```
vhdl2ta n_xor.vhd -t delay.tmp 2 -e env1.env -m sim -o testbench.vhd
-ts max
```

Le comportement attendu de la sortie de la porte dans ce test est donné comme suit :

```
s : 7 down, 14 up, 17 down, 24 up;
```

Le chronogramme obtenu par la simulation du programme VHDL temporisé généré en utilisant l'outil ModelSim est décrit dans la figure 4. Comme on peut le voir, le comportement du signal de sortie *s* dans est bien conforme à ce qu'on attend.

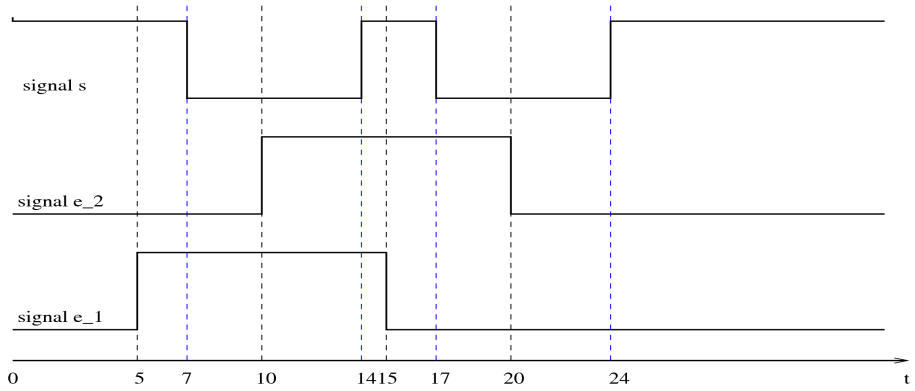


Figure 4 : le chronogramme des signaux d'entrée et de sortie obtenu par simulation (le premier test).

Le modèle temporel employé précédemment n'est pas complet car il associe deux délais ponctuels pour la porte pour toutes les configurations. L'un d'entre eux pour le front montant et l'un autre pour le front descendant. Pour avoir un modèle complet et précis, on doit employer le fichier de timing associé à la porte qui donne pour chaque configuration son délai ponctuel exact. C'est le modèle que nous avons employé dans le deuxième test présenté ci-dessous.

Deuxième test :

Le nouveau fichier régénéré qui décrit la fonctionnalité de la porte n-xor est présenté dans l'annexe A.2 (le fichier testbench généré est toujours le même que celui du précédent). Le programme VHDL obtenu est plus précis maintenant car il intègre pour chaque configuration fonctionnelle son délai ponctuel propre à partir de ce nouveau modèle complet de délais.

La commande employée pour générer ce nouveau programme VHDL temporisé est donnée comme suit :

```
vhdl2tv n_xor.vhd -t signals_timing.tmp 2 -e env1.env -o
```

```
testbench.vhd
```

La commande `vhdl2ta` qu'on passe ici pour générer ce programme VHDL temporisé est donnée comme suit :

```
vhdl2ta n_xor.vhd -t signals_timing.tmp 2 -e env1.env -m sim -o
testbench.vhd
```

Le comportement attendu de la sortie de la porte cette fois est donné comme suit :

```
s: 7 down, 14 up, 16 down, 23 up;
```

Le chronogramme obtenu par la simulation du nouveau programme VHDL temporisé généré est donné dans la figure 5. Comme on peut le voir, le comportement de la sortie de la porte est bien conforme au comportement décrit ci-dessus.

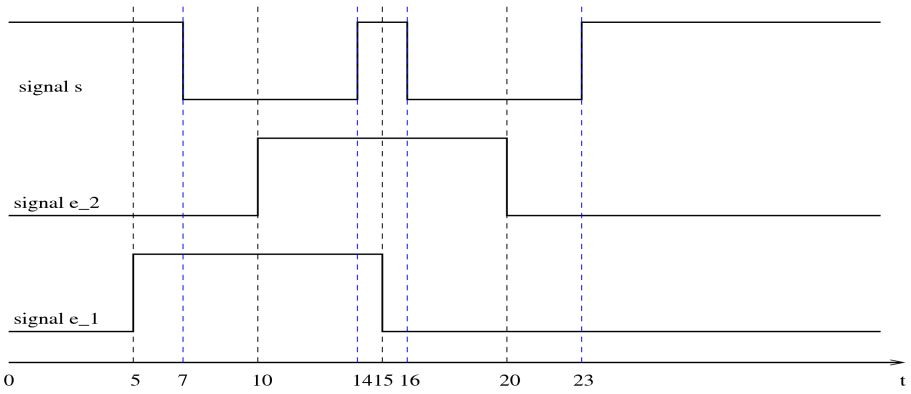


Figure 5 : le chronogramme des signaux d'entrée et de sortie obtenu par simulation (le deuxième test).

Comparaison avec le formalisme d'automates temporisés : Le modèle des programmes *VHDL* temporisés générés par l'outil VHDL2TV en utilisant le modèle temporel complet est plus précis contrairement au modèle d'automates temporisés. Cependant, la taille de ces programmes, qui dépendent du nombre de configurations des fichiers de timing associés aux portes du circuit à analyser, est assez grande. On rappel que dans le modèle d'automates temporisés générés par l'outil VHDL2TA, les délais des configurations ne sont pas spécifiés d'une manière précise (car ceci rend le modèle plus complexe et augmente beaucoup sa taille). En fait, pour tous les fronts montants (resp. descendants) de sortie de chaque porte du circuit à analyser, on prend en compte tous les délais compris dans l'intervalle *[dl+, du+] (resp. [dl-, du+])*, tel que *dl+* et *du+* (resp. *dl-* et *du-*) représentent respectivement le temps minimum et le temps maximum de tous les délais des configurations induisant un front montant (resp. descendant) sur la sortie de la porte. Comme conséquence, le deuxième modèle est une abstraction du premier.

5. Conclusion

L'outil VHDL2TV permet de générer des programmes VHDL temporisés modélisant les circuits à analyser par simulation, à partir de leur partie fonctionnelle décrite par un simple programme VHDL et leur partie temporelle. L'outil est testé sur divers exemples de circuits asynchrones (certains avec des mémoires). De plus, les programmes VHDL temporisés générés représentant des circuits à cent portes ont été analysés par l'outil de simulation ModelSim. C'est le cas de l'architecture complète abstraite de la mémoire SPSMALL de 3 mots de 2 bits, extraite au LIP6. Une présentation détaillée de l'analyse de cette dernière est donnée dans [BE11b – pp :66-83] et [BE10].

Références

[BE10] A. Bara, E. Encrenaz. Analyse de l'architecture abstraite de la mémoire SP-SMALL (3x2 bits), présentation VALMEM, Dec 2010.

[BE11a] A. Bara, E. Encrenaz. VHDL2TA : outil de traduction en automates temporisés des circuits décrits en VHDL – Manuel d'utilisation, rapport interne de l'equipe Soc – Lip6, 2011.

[BE11b] A. Bara, E. Encrenaz. VHDL2TA : outil de traduction en automates temporisés des circuits décrits en VHDL – Jeu de tests, rapport interne de l'equipe Soc – Lip6, 2011.

[BE11c] A. Bara, E. Encrenaz. VHDL2TV: http://www.lsv.ens-cachan.fr/~encrenaz/valmem/vhdl2tv/index.html, 2011.

[CEFX06] R. Chevallier, E. Encrenaz-Tiphene, L. Fribourg, W. Xu, *Timing Analysis of an Embedded Memory: SPSMALL*, WSEAS Transactions on Circuits and Systems, vol 5(7), pp 973-978, 2006.

[MS] ModelSim - Outil de simulation HDL : http://model.com/.

A. Exemples de circuits

A.1. Test1

Le programme VHDL temporisé généré par l'outil VHDL2TV dans le premier test est constitué de deux fichiers VHDL :

1. le fichier testbench donné ci-dessous.

```
-- Entity Declaration
entity test_n_xor is
end test_n_xor ;

-- Architecture Declaration
architecture arc of test_n_xor is

--Declaration des signaux (equipotentielles) internes.
-- entrees.
        signal e_1 : bit ;
        signal e_2 : bit ;
-- sorties.
        signal s : bit ;

-- déclaration des composants.
        component n_xor is
                port
                (
                  e_1 : in bit ;
                  e_2 : in bit ;
                  s : out bit
                ) ;
        end component n_xor ;

begin

-- Instanciation du circuit n_xor.
        inst_comp_n_xor: n_xor port map (e_1, e_2, s) ;

-- le processus associé à l'environnement.
        env: process
        begin
          e_1 <= '0' ;
          e_2 <= '0' ;
          wait for 100 ns ;
          wait for 5 ns ;
          e_1 <= '1' ;
          wait for 5 ns ;
          e_2 <= '1' ;
          wait for 5 ns ;
          e_1 <= '0' ;
          wait for 5 ns ;
          e_2 <= '0' ;
          wait for 120 ns ;
           assert false report "fin de simulation" severity failure;
        end process ;
end arc ;
```

2. le fichier décrivant la fonctionnalité de la porte n-xor donné ci-dessous.

```
-- Entity Declaration

entity n_xor is
        port
        (
          e_1 : in bit ;
          e_2 : in bit ;
          s : out bit
        ) ;
end n_xor ;

-- Architecture Declaration
architecture arc of n_xor is

begin

        pr_s: process (e_1, e_2)
          variable s_bis : bit ;
        begin
          s_bis := not (e_1 xor e_2) ;
          if NOW = 0 ns then
            s <= s_bis after 0 ns ;
          else
            if s_bis = '1' then
              s <= '1' after 4 ns ;
            elsif s_bis = '0' then
              s <= '0' after 2 ns ;
            end if ;
          end if ;
        end process ;
end arc ;
```

A.2. Test2

Le programme VHDL temporisé généré dans le deuxième test est constitué de deux fichiers VHDL :

1. le fichier testbench qui est le même que celui du premier test.

2. le fichier décrivant la fonctionnalité de la porte n-xor donné ci-dessous.

```
-- Entity Declaration
entity n_xor is
        port
        (
          e_1 : in bit ;
          e_2 : in bit ;
          s : out bit
        ) ;
end n_xor ;

-- Architecture Declaration
architecture arc of n_xor is

begin
        pr_s: process (e_1, e_2)
           variable s_init : boolean ;
        begin
           . . .
           . . .
           . . .
        end process;
end arc ;
```

```
s_init := false ;
if NOW > 0 ns then
  if e_2'event and e_1 = '0' and e_2 = '1' then
    s <= '0' after 2 ns ;
  elsif e_2'event and e_1 = '0' and e_2 = '0' then
    s <= '1' after 3 ns ;
  elsif e_1'event and e_1 = '1' and e_2 = '0' then
    s <= '0' after 2 ns ;
  elsif e_1'event and e_1 = '0' and e_2 = '0' then
    s <= '1' after 3 ns ;
  elsif e_2'event and e_1 = '1' and e_2 = '0' then
    s <= '0' after 1 ns ;
  elsif e_2'event and e_1 = '1' and e_2 = '1' then
    s <= '1' after 4 ns ;
  elsif e_1'event and e_1 = '0' and e_2 = '1' then
    s <= '0' after 1 ns ;
  elsif e_1'event and e_1 = '1' and e_2 = '1' then
    s <= '1' after 4 ns ;
  end if ;
else
  s <= not (e_1 xor e_2) ;
end if ;
```

B. Les grammaires des descriptions d'entrée de VHDL2TV

B.1. La grammaire des descriptions fonctionnelles en VHDL

```
vhdl_file    : entity architecture ;

entity       : ENTITY VHDLID IS PORT '(' ports ')' ';' END [VHDLID] ';'  ;

ports        : port | ports ';'  port ;
port         : VHDLID ':' mode BIT ;
mode         : IN | OUT  ;

architecture  : ARCHITECTURE VHDLID OF VHDLID IS def_signals BEGIN statements END
                [VHDLID] ';' ;

def_signals : def_signal | def_signals def_signal ;
def_signal  : SIGNAL VHDLID ':' BIT ';'  ;

statements  : Ɛ | statements signal_assignment_statement | statements process_statement ;

signal_assignment_statement   : VHDLID "<=" expr1        ;

expr1        : expr |  expr op2 expr ;
expr         : BITVALUE | VHDLID | '(' expr op2 expr ')' | NOT expr  | '(' expr ')' ;
expr         : VHDLID '=' BITVALUE ;

process_statement : process_label ':' PROCESS '(' signals_names_list ')' BEGIN
                     process_statement_part END PROCESS [process_label] ';'
                    ;

process_statement_part : if_statement ;
if_statement : IF if_statement_ END IF ';' ;
if_statement_ :   condition THEN signal_assignment_statement ELSIF if_statement_
                | condition THEN signal_assignment_statement
                | condition THEN signal_assignment_statement ELSE signal_assignment_statement
                ;

condition    : guard_expr1  ;
guard_expr1: guard_expr |  guard_expr op2 guard_expr ;
guard_expr        :  VHDLID '=' BITVALUE
                  |  '(' guard_expr op2 guard_expr ')'
                  |  NOT guard_expr
                  |  '(' guard_expr ')'
                  ;

signals_names_list : signal_name | signals_names_list ',' signal_name ;

process_label : VHDLID ;
```

signal_name : VHDLID ;

op2 : AND | OR | XOR ;

BITVALUE : '0' | '1' ;
VHDLID : Id ;
FILE_NAME : Id2 ;
NUMBER : Nat ;

Les mots présentés ci-dessous sont des mots clés. Elles peuvent être s'écrire aussi en minuscules.

ENTITY ARCHITECTURE OF IS BEGIN END SIGNAL BIT IN OUT PROCESS IF ELSE ELSIF THEN NOT AND OR XOR.

B.2. La grammaire des descriptions temporelles

Comme on a vu auparavant, il existe deux types de descriptions temporelles que l'outil VHDL2TA prend en entrée :

1. les descriptions d'intervalles des délais associés à chaque porte du circuit à modéliser (donné sous forme d'un fichier).

2. les descriptions dans lesquelles on associe un fichier de timing (configuration, front-sortie, temps) pour chaque porte du circuit. Ces fichiers sont accessibles via les informations mentionnées dans un fichier principale (représentant de la description temporelle).

Ci-dessous, on présente les deux grammaires associées à ces deux modèles.

B.2.1 La grammaire de la syntaxe du premier modèle :

temp_file : delay_signals ;

delay_signals : delay_signal | delay_signals delay_signal ;
delay_signal : signal ':' time_front_dn ',' time_front_up ';'
| signal ':' time_front_up ',' time_front_dn ';'
;

time_front_dn : time dn | '[' time ',' time ']' up ;
time_front_up : time up | '[' time ',' time ']' dn ;

signal : VHDLID ;
time : NUMBER ;
up : UP | '+' ;
dn : DOWN | DN | '-' ;

B.2.2. La grammaire de la syntaxe du deuxième modèle :

Grammaire A (fichier principale) :

temp_file : temp_signals ;

temp_signals : temp_signals temp_signal | temp_signal ;
temp_signal : signal timing_file_signal ;

signal : VHDLID ;
timing_file_signal : FILE_NAME ;

Grammaire B (fichier de timing de portes) :

time_file : sorties ;

sorties : sorties sortie | sortie ;

sortie : param configurations ;

param : signals_in out_in TIME ;

signals_in : signal | signals_in signal ;

configurations : configuration | configurations configuration ;

configuration : n_io time

n_io : n_io io | io ;
io : '0' | '1' | dn | up ;

signal : VHDLID ;
time : NUMBER ;
up : 'u' | 'U' ;
dn : 'd' | 'D' ;

VHDLID : Id ;
FILE_NAME : Id2 ;
NUMBER : Nat ;

Les mots UP, DOWN, UP, TIME, sont des mots clés. Elles peuvent être s'écrire aussi en minuscules.

B.3. La grammaire des descriptions d'environnement :

env_file : type_env behaviors_env ;

type_env : [CYCLIC] AUTOMATON ENVIRONMENT
| [AUTOMATON] CYCLIC ENVIRONMENT
| AUTOMATA ENVIRONMENT
| Ɛ

;

behaviors_env : [clock_def] signals_behavior ;

clock_def : CLOCK signal WITH THI time TLO time [INVERSE] [NCYCLES NUMBER] ';' ;

signals_behavior : signal_behavior | signals_behavior signal_behavior ;

signal_behavior : signal ':' time_fronts ';'
| signal ':' value ';'
;

time_fronts : time_front | time_fronts ',' time_front ;

time_front : time front | '[' time ',' time ']' front ;

front: up | dn ;
signal : VHDLID ;
time : NUMBER ;
value : '0' | '1' ;
up : UP | + ;
dn : DOWN | DN | '-' ;
VHDLID : Id ;
NUMBER : Nat ;

Les mots CYCLIC, AUTOMATON, ENVIRONMENT, CLOCK, WITH, THI, TLO, INVERSE, NCYCLES, UP, DOWN, DN, sont des mots clés. Elles peuvent être s'écrire aussi en minuscules.

Id2 : Id2_ Id | Id2_ Id(.Id)
Id2_ : Id2_ Id '/' | Ɛ ;
Id : Alpha | Id AlphaNum ;
Nat : Num | Nat Num ;
Alpha : A | . . . | Z | a | . . . | z ;
Num : 0 | . . . | 9 ;
AlphaNum : Alpha | Num | _ ;